Glorious Strings

Part (a): Working with a partner, figure out what is displayed when the Python code is executed. Consult the “Strings Cheat Sheet,” as needed. The diagrams are provided to help you keep track of indices. (You could use a strategy like this on an exam to reduce the chance of errors due to miscounting.)

```
word = 'telephone'
print( word[1] )
print( word[1:4] )
print( word.find('ph') )
print( word.replace('e','oi') )
print( word [-2::2] )
word = 'spot'
print( word[1:] )
print( word + 'ty' )
print( word[:len(word)] )
print( word[:-1][-1] )
print( word[:word.index('p')] + word[word.index('p')+1:] )
print( word[:word.find('e')] )
print( (2*word)[::2] )
print( word[:-2].upper() )
r = ''
for c in word:
r = c + r
print ( word, r )
```

To check you answers, step through Visualization 0 using the link in the Artifacts section of the website. Ask about any answers that you don't understand.
Part (b): Tracing execution of an algorithm on a specific example can help you figure out what the algorithm does in general. To trace an execution, you list the line numbers of the statements that are executed, in order, and show the values of any variables that are changed at each statement.

Finish the following trace of mystery1.py assuming the user enters 'Cu L8r!' at the prompt (without the quotes).

<table>
<thead>
<tr>
<th>Line no.</th>
<th>Variable(s) changed</th>
<th>Line no.</th>
<th>Variable(s) changed</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>w: 'Cu L8r!'</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>r: ''</td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>c: 'C'</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

When done, step through Visualization 1 to check your trace. (For PythonTutor to simulate execution of the input expression, you will need to enter Cu L8r! and then press submit in the box below the progress bar.)

What will be printed if the user enters 'I do not like them, Sam-I-Am.' at the prompt?

What will be printed if the user just hits the enter/return-key at the prompt?

How is the output of the program related to the input?
Part (c): Finish the following two traces of mystery2.py. For the first trace, assume the user enters "Endive" at the prompt (without the quotes); for the second, assume the user enters "Madam".

<table>
<thead>
<tr>
<th>Line no.</th>
<th>Variable(s) changed</th>
<th>Line no.</th>
<th>Variable(s) changed</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>w: &quot;Endive&quot;</td>
<td>4</td>
<td>w: &quot;Madam&quot;</td>
</tr>
<tr>
<td>6</td>
<td>e: 5</td>
<td>6</td>
<td></td>
</tr>
</tbody>
</table>

When done, step through Visualization 2 to check your trace. (Once you provide the user input in a visualization, PythonTutor allows you to “explore” the execution by stepping both forwards and backwards through the visualization or restarting from the beginning of the visualization. But restarting does not allow you to provide new user input. To provide different user input (e.g., to test using "Madam" after using "Endive"), you need to press the Edit Code link and then, in the editor, press the Visualize Execution button.)

- What will be printed if the user enters "A man, a plan, a canal: Panama" at the prompt?
- What will be printed if the user just hits the enter key at the prompt?
- For what inputs does the program print "Bingo!"?
- Download mystery2.py and bring it up in Spyder. Modify this program so that it also prints "Bingo!" for input like the one in the second bullet, in which the letters form a palindrome when spaces and punctuation are ignored.
**Part (e):** In this part, you will work with a partner to write a program that prompts the user for a book title and displays a version of the book title that follows certain capitalization rules for titles. Because the purpose is gain experience with string operations and control flow, your program will use only string operations, control statements, assignment statements, input statements and print statements (like this week’s CSE 231 project, you are NOT to use lists!). You will develop the program incrementally, as described below.

**Version 1:** To apply any capitalization rules, your program will first need to be able to isolate the words in the title. So the first version of the program will just implement the following algorithm:

```plaintext
-- prompt the user for a book title
-- loop:
  * assign the next word of the title to a variable
  * print the value of this variable on its own line
```

You can assume that at least one a space separates consecutive words in the title entered by the user. You should treat any punctuation as part of the word it is attached to.

To clarify, some sample interactions with our first program are shown below. (The first line shows the prompt printed by the input expression and the title entered for testing.)

![Console output](image1)

![Console output](image2)
**Version 2:** For this version, display a title in which every word begins with a capital letter and a space separates consecutive words. Do this by adding code to:

* Assign the empty string to a new variable (a ‘working variable’)
* On each iteration of the loop, update the working variable to include a capitalized copy of the new word and a space character, as needed
* Print the fully capitalized title with exactly one space between words.

For example:

```
Enter a title: gone with the wind
Gone With The Wind
```

```
Enter a title: Oh, the places you'll go!
Oh, The Places You'll Go!
```

**Version 3:** Certain short words should be capitalized in a title only if they are the first or last word in the title. Modify your program so that it capitalizes the following short words if and only if they are first or last in the title.

Short words not to capitalize: a, an, and, the, but, for, on, at, to, from, by, with, it.

Some examples:

```
Enter a title: Gone With The Wind
Gone with the Wind
```

```
Enter a title: Oh, the places you'll go!
Oh, The Places You'll Go!
```

**Version 4:** There is an exception to the rule about short words. The first word following a colon (:) should always be capitalized. Modify your program to enforce this rule.

For example:

```
Enter a title: Berkeley UNIX: a simple and comprehensive guide
Berkeley UNIX: A Simple and Comprehensive Guide
```